APPLICATION FOR CAPTURING POTENTIAL TARDINESS IN OPERATING ROOM SUITES

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Bethlehem Abera Gronneberg

In Partial Fulfillment
For the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

March 2012

Fargo, North Dakota

North Dakota State University
Graduate School

APPLICATION FOR CAPTURING THE POTENTIAL FOR TARDINESS IN OPERATING ROOM SUITES

By

BETHLEHEM ABERA GRONNEBERG

The Supervisory Committee certifies that this ***disquisition*** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

MASTER OF SCIENCE

SUPERVISORY COMMITTEE:

Dr. Kendall Nygard
Chair

Dr. Tariq King

Dr. Gursimran Walia

Dr. Limin Zhang

Approved:

3/22/2012	Dr. Kendall Nygard
Date	Department Chair

ABSTRACT

This paper describes the design, architecture and implementation of a Dashboard Application, a tool developed to capture the potential tardiness in Operating Room (OR) Suites. The paper is focused on developing an automated application which is intended to monitor and display the progress of multiple out-patients on the day of their elective surgery as they advance from admission to the pre-operative stages leading up to their final destination of the operating room. The application will display OR patients' real-time location, the time they entered at that location and the time remaining for their scheduled surgery once they arrive at the admission office.

The objective of this paper is to implement an automated application using real-time locating technology to provide timely and accurate information regarding patient preparedness for surgery. This information otherwise would have taken the traditional modes of communication (face-to-face and telephone) to check for patient preparedness.

ACKNOWLEDGEMENTS

This work would not have been possible without the continued guidance and direction of Dr. Kendall E. Nygard, my paper advisor.

I am thankful to each of the members of my advisory committee, Dr. Tariq King, Dr. Gursimran Walia and Dr. Limin Zhang for their valuable time and valuable recommendations in preparing this master's paper.

I am especially grateful to Dr. Raymond Gruby and Mr. Trevor Gruby, my co-workers at Intelligent InSites, for their invaluable input and support during my research topic discovery phase.

I enjoyed immense encouragement from members of my family while pursuing my graduate school experience. I would like to thank my parents whose continued words of prayer provided me with comfort and strength. My husband, Ron, whose love and unfailing support are like no other. A very special love and thank you goes to my three wonderful sons - Joshua, Gabriel and Nathaniel, who gave me unending cheer and inspiration to keep me motivated.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDIX TABLES

1. INTRODUCTION

1.1. The Background

A healthcare system constitutes the care providers along with the institutions and the resources necessary for the development and delivery of quality services. According to the 2011 report of the Organization for Economic Co-operation and Development (OECD), the United States (US) leads the rest of the world in healthcare expenditure, costing more per person than any other nation [1]. This rising cost, together with the healthcare institutions' claims of lower reimbursement rates, higher costs of liability insurance, drugs, medical devices and other supplies, has been forcing US hospitals to find better ways to improve their slow growth of productivity and efficiency.

Currently, there are a number of initiatives targeted to better manage productivity and efficiency in the healthcare industry [2]. One initiative is geared toward increasing efficiency in expensive Operating Room (OR) suites, using the technology and infrastructure that already exists in the facility. The concept of increasing the number of surgical cases performed on any given day while keeping the cost of resources to a minimum is not only crucial but also a very complex undertaking. This complex undertaking has numerous benefits in addition to the reduction of operational costs resulting from proper utilization. Achieving higher throughput is directly related to the decrease in patient delays and the increase in satisfaction among patients, staff, and physicians [3].

The factors contributing to the inefficiency of a surgical suite include but are not limited to, misallocation of staff to workloads, start-time tardiness, recovery admission delays, cancellation costs and turn-over times [4].

This research work identifies start-time tardiness as one of the key contributors to the inefficient utilization of OR suites. Start-time tardiness is quantified as the difference between the time the patient actually enters the OR for surgery and the scheduled start-time of the case on a given day. Tardiness from scheduled start-times is a cause for dissatisfaction of patients and a source of frustration for operating room personnel [5]. There have been various studies conducted in identifying the factors affecting efficiency in OR suites in general and reducing tardiness in the first case of the day in particular [6,7,8,9].

1.2. The Motivation

This paper is motivated by the need to mitigate the effects of manual communication by streamlining the coordination and communication of resources between OR suites and other healthcare departments.

The task of coordinating resources to better manage all aspects of a surgical suite requires constant communication and coordination between various hospital units and healthcare workers from diverse disciplines [7]. Due to the complex nature of this task, healthcare facilities hire OR managers to coordinate preparedness of patients for surgery, negotiate time for new surgical cases, reschedule delayed or cancelled surgical cases, ensure the cleanliness of operating rooms and the preparedness of appropriate equipment for the planned surgeries, and ensure that well-suited operating room staff are assigned.

In healthcare facilities that do not have OR managers, the added assignment of making managerial decisions falls on the shoulders of anesthesia providers and/or OR charge nurses throughout the day. Charge nurses communicate by phone, intercom or face-to-face with other hospital units, such as the patient admitting and holding areas, out-patient surgery units and the

Post Anesthesia Care Unit (PACU). Findings from a previous research study by Moss et al. reveal that the most frequent mode of communication between OR charge nurses and other hospital units occurred face-to-face (69% of the time), followed by phone (18% of the time) and intercom (7% of the time). The communication was related to staffing, equipment management and patient preparedness for surgery [7]. The face-to-face communication is further challenged if the physical location of the point of entry (admission) for patients is located on a different floor or hospital wing from that of the OR suites and PACU [8].

In high traffic and interrupt-driven environments such as OR suites, the accuracy and timeliness of the information exchange regarding the preparedness of patients for surgery is negatively affected. When time is the subject of communication during such cases, 46% of interactions can involve high tension and blame [9]. This paper is motivated by the need to provide better visibility to the progress of patients as they progress through the pre-operative stages and capture potential start-time tardiness for scheduled surgical cases in OR suites. This paper is also motivated by the need to automate the information delivery mechanism to provide timely updates to charge nurses.

1.3. The Objective

The objective of this paper is to implement an automated application using real-time locating technology to provide timely and accurate information regarding patient preparedness for surgery. This information otherwise would have taken the traditional modes of communication (face-to-face and telephone) to check for patient preparedness.

This paper is focused on developing an automated tool which is intended to monitor the progress of multiple out-patients on the day of their elective surgery as they advance from

admission to the pre-operative stages leading up to their final destination of the operating room. The application will track OR patients and displays their real-time location, the time they entered at that location and the time remaining for their scheduled surgery once they arrive at the admission office.

By providing an instance access to monitored movement on a dashboard in OR suites, charge nurse will have real-time and accurate information to assist making proactive real-time decisions to mitigate the potential start-time tardiness. This information otherwise would have taken the traditional modes of communication (face-to-face and telephone) to check for patient preparedness.

The rest of this paper is organized in the following way: Chapter 2 discusses related studies conducted with regard to reducing start-time tardiness and improving OR suite efficiency. This will be followed by Chapter 3 illustrating the approaches and methods employed to reduce tardiness using the Real Time Locating System (RTLS). The last two chapters describe the result of the study and the future interests of this paper.

2. RELATED WORK

2.1. The Background

Table 2.1 defines important terminologies relevant to this paper to better understand the healthcare services interpretations.

Table 2.1 Definition of Terms Used in This Paper

Term	Definition
Surgery	Operative procedure in a hospital setting that treats diseases or disorders by performing cutting, removing or changing part of the human body.
Elective Surgery	Surgery that is scheduled in advance because it does not involve a medical emergency. It is a surgery beneficial to the patient but not essential for survival.
Outpatient Surgery	Surgery that does not involve a hospital stay. This service can also be called ambulatory surgery, same-day surgery or day surgery.
Operating Room Suite	Facility within a hospital where surgical operations are performed in a sterile environment. An operating room suite may include several operating rooms, scrubbing corridors, pre- operation and recovery areas, offices, a storage and cleaning facility, and any other unit necessary for conducting multiple surgeries.
Charge Nurse	Nurse assigned to manage the operations of the operating room suite

Table 2.1 (continued)	
	for the shift. A charge nurse is responsible for the staffing, admissions and discharge, and coordination of activities in the patient care area.
List of Cases	List of patients' procedures that a surgeon or surgeons are scheduled to perform sequentially in an OR suite; organized and issued for the day.
Start-Time	Defined time where a scheduled case should enter the OR to start the surgery.
Tardiness	Lapse of time incurred from a scheduled start-time. If a case is scheduled to enter its designated OR at 10:00 AM but enters at 10:40 AM, then that case is tardy by 40 minutes.
Turnover Time	Time incurred between surgery cases in an OR. The time when the previous case is done and the new case arrives.
Patient Flow	Movement of patients through a set of locations in a healthcare facility during the pre-operative stages in preparation for surgery.
Case Duration	Time it takes for a specific surgical case to be completed.
Case Duration Bias	Indications of whether the scheduled estimate of case duration is consistently too high or consistently too low. Prediction bias in case duration is expressed as estimates per 8 hours of OR time.
Under-Utilized or	Under-utilized time represents the hours for which staffing have

Table 2.1 (continued)	
Over-Utilized OR	been planned but the OR sits idle. Over-utilized time represents the hours that cases extend beyond the length of their allotted time for which staffing has been planned in the OR [5].
Real-Time Locating System (RTLS)	System that uses sensor technology to track and locate people or assets in real time.

2.2. Related Work

This chapter examines and summarizes related software modules to this paper's proposed solution – the Dashboard Application.

2.2.1. Skytron Asset Manager

The Skytron Asset Manager is a patented wireless Radio Frequency ID (RFID) solution which enables medical centers to locate and track medical equipment real-time. Skytron Asset Manager enhances equipment utilization and has the ability to proactively alert Biomedical and Materials Management staff when equipment par levels fall below safety stock limits, when rental equipment contracts are about to expire, or when it is time to perform scheduled Preventative Maintenance. Robust reporting suite provides hospital administrators real-time actionable information to effectively improve business processes that impact patient care and operational efficiency.

Like Skytron Asset Manager, this paper's proposed solution, the Dashboard Application is designed to use Real-Time Locating System (RTLS) in healthcare facilities. Unlike Skytron

Asset Manager, however, the Dashboard Application is used to track and monitor out-patients scheduled for Operating Room as they undergo preoperative procedures in preparation for surgery.

2.2.2. The RealView Solution

The RealView Solution of a company called PeriOptimum claims to maximize the capacity of OR suites by communicating case status to physicians, nurses, and staff aided by a visual interface. The solution also provides a real-time feed back to the family members in the waiting room and lounges regarding the progress of patients inside the operating room.

Compared to the RealView, this paper's Dashboard Application addresses one particular aspect of a patient tracking RTLS system, providing visibility to patient progress and capturing the potential for tardiness. The Dashboard Application will implement a way to provide visibility to the patient flow before the patient is wheeled into the OR room and before the tardiness stage is reached.

2.2.3. Microsoft Project Scheduling

Microsoft Project is a window-based project management and scheduling software program, developed by Microsoft. It is designed to assist a project manager in developing a plan, assigning resources to tasks, tracking progress, managing the budget, and analyzing workloads. As resources (people, equipment and materials) are assigned to tasks and assignment work estimated, the program calculates the cost, equal to the work times the rate, which rolls up to the task level and then to any summary tasks and finally to the project level. Resource definitions can be shared between projects using a shared resource pool.

Similar to Microsoft Project, the Dashboard Application can be used as an effective tool to track the progress of patients, assist the charge nurses to make informed decisions in planning resource allocation and negotiation of new and re-scheduled surgical cases. Besides the fact that the Dashboard Application is a real-time monitoring of patient movements, its main differentiator from Microsoft Project software is that the Dashboard Application is a web based application that can be accessed from any type of computer without installing software on user's computer. The Dashboard Application is an event-driven passive status display that doesn't require user interaction to view and operate it.

3. METHODS

3.1. The Benchmark

In high traffic and interrupt-driven environment such as the OR suites, the accuracy and timeliness of the manual information exchange is prone to error [7]. Overhead in communication styles, lack of visibility and untimely coordination with other hospital units pertaining to patient preparedness for surgery can be the potential to start-time tardiness of scheduled surgical cases.

Findings of the study by Moss et al reveals that the most frequent mode of communication patterns between OR charge nurses and other hospital units occurred face-to-face (57% of the time), followed by telephone and occasional use of intercom [7]. The purpose of the communication is focused on negotiating time for a new case (patient), negotiating a change in case time due to late arrival, cancelling a case, assigning a case to a particular room, assigning particular staff to a case or arranging for staff coverage during breaks.

3.2. The Premise

This paper states the following premise. Building an automated Dashboard Application that continuously updates the information about the scheduled and arrived patients' pre-surgical status and presenting it visibly on monitors similar to the airport display screens with current flight status will have the following benefits:

- Provides visibility to the communication aspects of patient preparedness for surgery so that hospital staff don't have to use traditional modes of communication (face-to-face and telephone).

- Provides real-time and accurate information to assist the charge nurse to make proactive real-time decisions.
- Helped capture potential areas for tardiness.

This paper will support the above outlined premise using implementation of the Dashboard Application tool and conducting a simulation of scenarios for a set of sample patients.

3.3. Components of Dashboard Application

3.3.1. The Background

Real Time Locating System (RTLS) is a location sensing technology used to find and track objects or people in real time. It is used to monitor and analyze the information acquired on the movements of located entities (such as patients, staff or equipment). This paper focuses on tags used to track patients in healthcare facilities.

Several technologies are used to build up RTLS sensors. Some use dedicated Radio Frequency Identification (RFID) tags and readers while others use existing Wireless Local Area Network (WLAN) networks and add RTLS ability to those networks. Still others use light, camera vision, infrared, sound, ultrasound, Bluetooth, Wi-Fi, ZigBee, Ultra Wideband, Global Positioning System (GPS), Cellular, hybrid systems and many more technologies [11]. Different technologies use different approaches and each approach solves a slightly different problem or supports different applications and provides different levels of precision in positioning identification.

The components of RTLS are [12]:-

- **Tags** – small portable and wireless devices that have built-in Infrared (IR), Wi-Fi, etc technology. These battery-operated tags get clipped on a patient's clothing for the purpose of tracking. Each tag emits a signal carrying that tag's serial number. The batteries of these tags are designed to be either rechargeable or replaceable. The life of a battery depends on its usage. The average battery life is shorter for active RTLS tags that are constantly in motion than for the tags programmed to only transmit at shorter, predefined intervals.
- **Location sensors** – devices mounted on various rooms, hallways and corridors to sense the movement of tags using location movement-sensing technologies such as RFID, Ultra Sound, etc. These devices are used to compute the range between various nodes in the system. The wireless sensors are usually deployed as a matrix of locating devices installed at a spacing of anywhere from 50 to 1000 feet. The location sensors flood these entire areas with their IR signals, capturing and transmitting frequent Giga-Hertz (GHz) information containing the tag's serial number it came across, as well as the location sensor's own unique ID number.
- **Location engines** – software used to calculate the patient's location by communicating with the location sensors that collected the received signals of that individual's tag and sensor's serial numbers (such as IR sensors). The location engine then runs an algorithm to determine where the tags are located.

- **Middleware -** software that resides among the pure RTLS technology components (the tags, the location sensors, and the location engine) and the business applications capable of exploiting the value of the technology.

- **Application Programming Interface (API)** - software that interacts with the RTLS middleware, analyzes and translates the data.

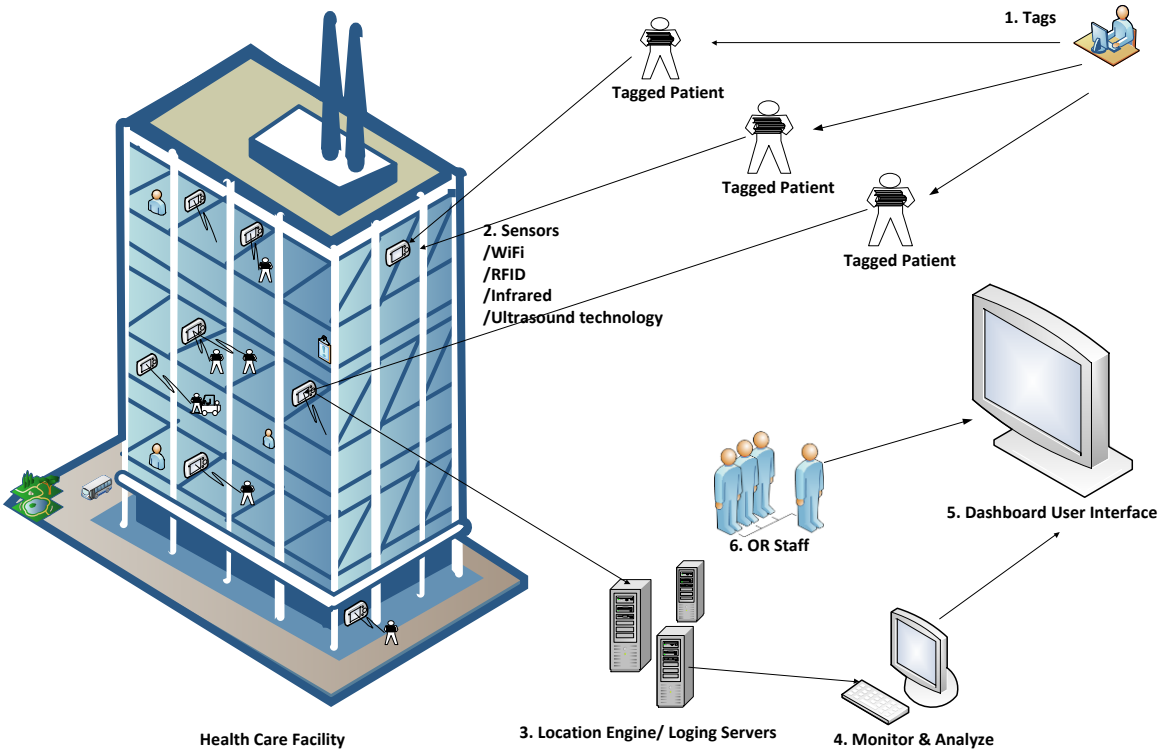

Figure 3.1 Components of RTLS and Communication Flow

Figure 3.1 demonstrates the components that make up the RTLS and the flow of communication and data exchange in a hospital setting. It starts with the tag assignment to patients at admission, followed by the tracking of tags with location sensors mounted in rooms, hallways and exit points. The location engines process the data transmitted by the location sensors, determining the exact location of the tags.

To provide the greatest flexibility, an RTLS platform should offer an open API so that it is capable of providing location and status data to both end-users and to third party applications. Fully interoperable RTLS should be able to seamlessly integrate with other software systems which can be enriched with real-time location information.

The analyzing and translating data using the API phase sorts out the tags and its association with patients and creates a database of location movements. The Dashboard Application makes an AJAX call to the location movement API that interprets the pure RTLS movement data. The application script then parses the returned data from the API as JSON (Javascript Object Notation). A visually interactive User Interface (UI) is built to display the resulting output that OR staff and other interested staff can visualize and determine the necessary follow-up.

Once a healthcare facility adopts the monitoring and reporting of real-time location systems and amasses the location information of its patients as they advance in their preoperative stages, this information can then be used to communicate to OR staff and interested hospital departments to better facilitate its operations and organization. The Dashboard Application will be such a tool to filter out only the scheduled OR patients' current location, time entered at that location and duration of time left before their scheduled surgery stars. The monitoring stops once the patients are wheeled into their destined OR suites. The Dashboard Application then interprets the result on monitors in OR suites, pre-operative areas and nurse stations to display the real-time movement of patients.

Figure 3.2 explains the layered technology that the Dashboard Application is built up-on. This technology uses REpresentational State Transfer (RESTful) web services to initiate request

to a server, query data and interpret the resulting data. It uses a third party Application Programming Interface (API) architecture which in turn aggregates and analyzes location data from location engines and middleware through various mounted sensory devices and RTLS tags. RESTful web service is a software architectural style consisting of clients and servers principles that use the Web as a platform for distributed computing.

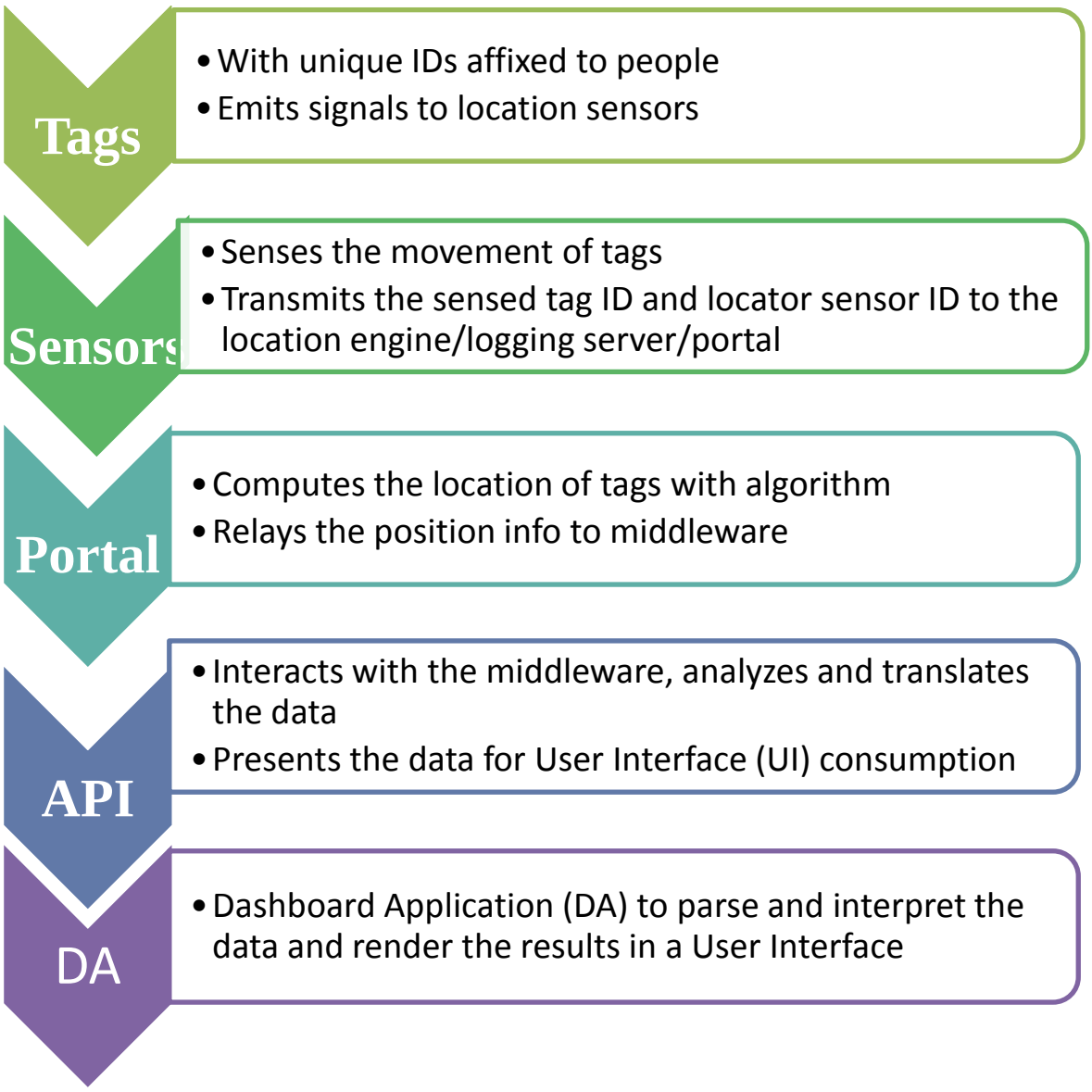

Figure 3.2 Layered Technology that Make up the Dashboard Application

Requests and responses are built around the transfer of representations of resources. Resources, which are conceptual objects such as entries in a database, are exposed via a uniform interface. On the web, this uniform interface would be the HTTP Protocol and the five standard well-known HTTP verbs: GET, POST, PUT, DELETE and OPTIONS. Representations of the state of the resource are accessed and operated on via the uniform interface. For example, to view the user account you GET a representation of its state; to update it you PUT a new representation; to remove it you DELETE the resource; etc.

Figure 3.3 Patients Movements in Pre-operative Milestones

Figure 3.3 is a screen shot of the Dashboard Application with different sets of scenarios playing real time. The application tries to answer one important question – is the patient ready for surgery? The traffic light color indicator icons are designed to give a quick visual indicator of the status at a particular location. The color indicator icon along with the duration of stay at that location will give the necessary information for the hospital staff to take the next action necessary.

The Dashboard Application is developed in such a way that it can be streamed through monitors mounted on a wall or accessed using mobile devices such as iPads. Using the RESTful application programming interface (API) architecture, the application consumes sensed data from third party applications and provides a way to interpret the data in a meaningful way using the User Interface (UI). This Dashboard Application retrieves required data from the server through asynchronous AJAX call using XML HTTP Request object.

When patients arrive at the registration desk of the healthcare facility for their scheduled surgery, they will be provided with a unique tag to clip on their clothing. This assigned unique tag's serial number (id) is entered in the Electronic Medical Record (EMR) system along with the patient's healthcare data such as patient's name, scheduled OR name and scheduled time of the surgery. The EMR system publishes a record updated event to HL7. Health Level Seven (HL7) is a standard for exchanging information between medical applications. This standard defines a format for the transmission of health-related information [13]. Information sent using the HL7 standard is sent as a collection of one or more messages, each of which transmits one record or item of health-related information.

The third party application listens to HL7 events and monitors the movements of the assigned tag from the RTLS location engines. The Dashboard Application retrieves this unique tag identified by its serial ID associated with its unique patient data available for that particular healthcare facility. This is accomplished by making the AJAX call to the Patient Location API that interprets the pure RTLS movement data. The application script then parses the returned data from the API as JSON (Javascript Object Notation). Figure 3.4 explains such data flow using the http request to the server and the parsing and rendering of the data in the User Interface (UI). A visually interactive UI is built to display the resulting output that OR staff and other interested staff can visualize and determine the necessary follow up.

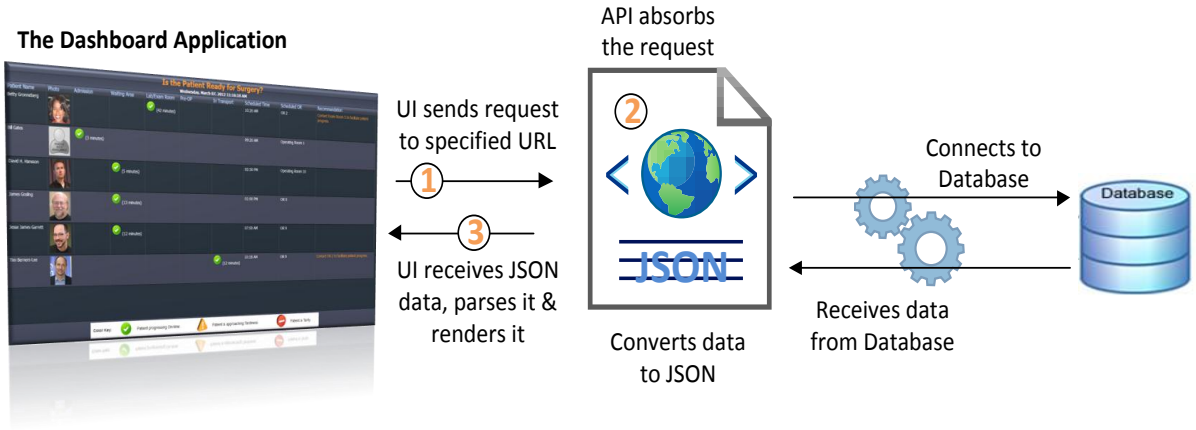

Figure 3.4 Dashboard Application System Data Flow

3.4. Dashboard Application Architecture

A good architecture is a necessary component for an application's scalability and maintainability [14]. The architecture is as much about providing structure and consistency as it is about actual classes and framework code.

The Dashboard Application uses a robust JavaScript framework with support for inheritance and encapsulation to develop graphically rich application that runs on all modern browsers and mobile devices. It leverages a client side design pattern called Model-View-Controller (MVC). The MVC paradigm is a way of breaking an application's interface, into three manageable parts: the model, the view, and the controller. MVC facilitates the implementation process by providing a logical partitioning to the objects involved in the design.

Figure 3.5 describes the Dashboard Application's architecture using the MVC pattern. After login, the Dashboard Application starts with an instance of application class. The application class contains global settings and maintains references to all of the models, views and controllers used by the application. The application class also contains a launch function,

which is run automatically when everything is loaded.

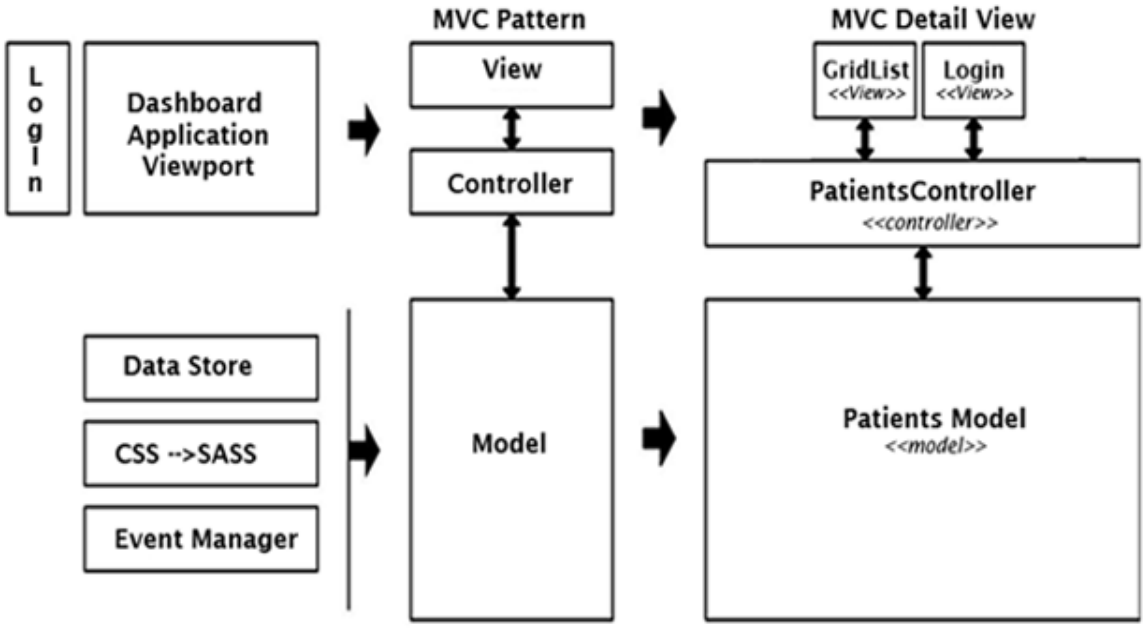

Figure 3.5 Dashboard Application MVC Pattern

The View (in this case the Dashboard Application table view) is simply a component that displays data contained in a store. A store can be thought of as a collection of records, or Model instances. The benefit of this setup is clear separation of concerns. The View is only concerned with displaying the data, while the Store is configured to fetch and save the data from the server using its Proxy.

The user logs into the system and the Dashboard Application's viewport builds the grid list that contains the data while the controller is set to listen to events regarding the patients' movements from the API. Figure 3.6 shows the code for instantiation and launching of the view (patientlist class) and the controllers (ApplicationController and PatientsController classes). Controllers are the glue that binds an application together.

```
Ext.Loader.setConfig({ enabled: true });
Ext.application({
    name: 'dashboardApp',

    launch: function() {
        Ext.create('Ext.container.Viewport', {
            layout: 'fit',
            items: {
                xtype: 'patientlist'
            }
        });
    }
    ,controllers: ['ApplicationController','PatientsController']

});
```

Figure 3.6 Code Snippet to the Dashboard Launch Application

All they really do is listen for events from views and initiate actions. Controllers have access to models, stores and views. Figure 3.7 shows the code for the Patient Model class which lists the fields that is used for parsing.

```
Ext.define('dashboardApp.model.PatientsModel', {
extend: 'Ext.data.Model',
fields:
    [
'id','uri',
{name:'patientName', mapping: 'name'},
{name: 'scheduledTime',
    mapping: function(n) {
        var patientScheduledTime;
        if(n.patient['patient-schedules']['total-count'] == 1) {
            patientScheduledTime = n.patient['patient-schedules'].value['start-date-time'];
        }
        return patientScheduledTime;
    },
    type: 'date', dateFormat: 'c'
},
{name: 'currentLocationName', mapping: "['current-location'].name"},
{name: 'dateEnteredCurrentLocation', mapping: 'date-entered-current-location', type: 'date', dateFormat: 'c'},
{name: 'scheduledOR', mapping: "patient['scheduled-location-names']"},
{name: 'mpi', mapping: "patient['master-patient-index']"}
,{name: 'photo', type:'ref'}
],
associations: [{type: 'belongsTo', model: 'dashboardApp.model.BinaryData', foreignKey: 'photo', getterName: 'getPhoto' }]
});
```

Figure 3.7 Code Snippet of the Patient Model Class

The grid list view is built by parsing the model that defines the necessary fields and processing the results returned from the AJAX call made by the store object which is listed in Figure 3.8. The store polls the API every 60 seconds for data in addition to location events.

```
Ext.define('dashboardApp.store.PatientsStore', {
    extend:'Ext.data.Store',
    model: 'dashboardApp.model.PatientsModel',
    autoLoad: true,
    proxy: {
            type: 'ajax',
            url: "/api/2.0/rest/patient-visits.json?word-separator=camel",
            extraParams: {
                    select:"type,current-location,patient,patient,photo,date-entered-current-location",
                    expand: "type,photo,patient.patient-schedules.scheduled-location,current-location",
                    filter: "current-location.name ne '' and current-location.name ne 'OR',
                    sort: "name ASC",
                    format: "json"
            },
            reader: {
                type:'json',
                root: "['list-response'].value"
            }
    }
})
```

Figure 3.8 Code Snippet for Patient Store Class

3.5. Sequence Diagram

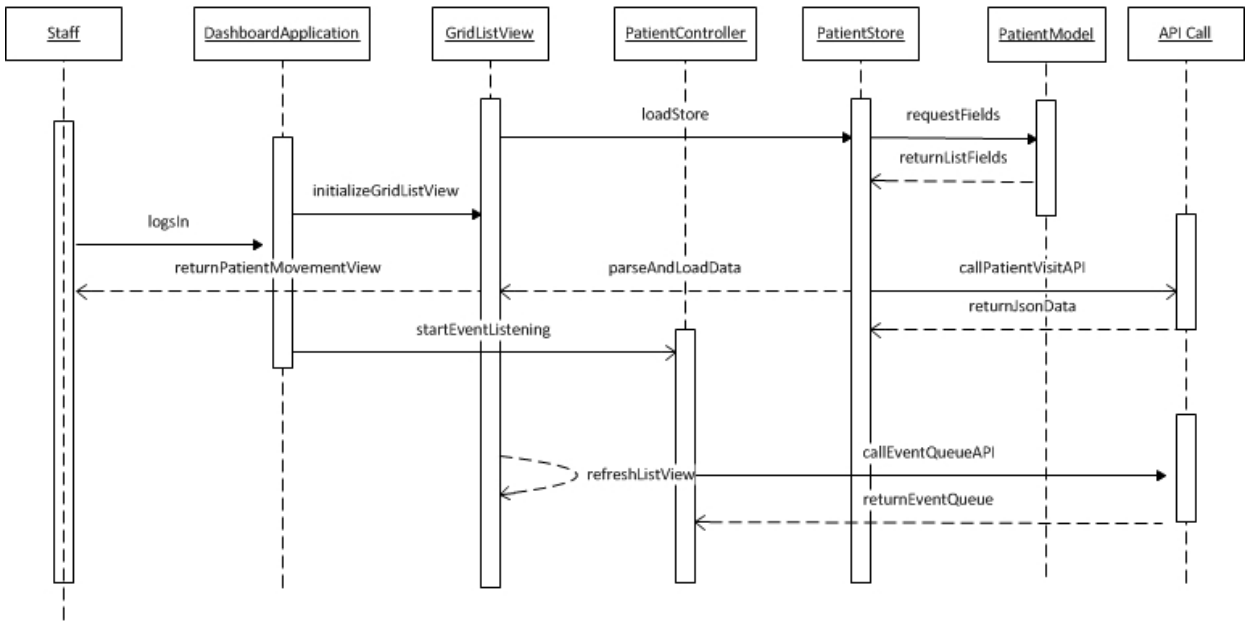

Figure 3.9 Sequence Diagram of the Dashboard Application

The main purpose of a sequence diagram is to define event sequences that result in a desired outcome. Figure 3.9 illustrates the interactions between the objects that make up the application's architecture. The use case described below in Table 3.1 captures these interactions.

Table 3.1 Use Case Description for Viewing Patient Movement

Actors	Admission, Pre-OP, OR, PACU staff
Use Case ID	DashboardApp-01-ViewPatientsMovement
Use Case Level	End-to-End
Details	The application gets launched with patient name and photo if the patient's photo is available, the current location and the duration at that location along with the scheduled start-time and name of the OR. An event listener gets registered for tracking patient movement and listening to new patient tagged event.
Pre-conditions	The user is logged into the system with proper credentials.
Description	• The dashboard application launches controller for event listening and invokes the Grid List View. • The Grid List View shall call the store to load the data. • The store uses the proxy to make an AJAX call to the API • The API returns the data as JSON • The store uses the model to parse the required fields • The Grid List View uses the store to consume and render the parsed data

Table 3.1 (continued)	
	• The controller sends an AJAX request for location moved Event • The controller passes the captured event to the store. • The store refreshes the in-memory data with the new data. • The Grid List View renders the parsed data. The use case ends when the data is displayed on the screen.
Post-conditions:	Staff monitors the movement of patients and takes the necessary actions.

3.6. Testing the Dashboard Application

There are many reasons to test applications. Tests can verify an application's functionality to eliminate the need to enumerate all the use cases manually. Also, if the application were to be refactored, or updated, the tests could verify that the changes did not introduce new bugs into the system.

3.6.1. Unit Testing With Jasmine

The Jasmine assertion library was used to write unit tests for the Dashboard Application. Jasmine is a framework for testing JavaScript code and can be run anywhere JavaScript is executed: a static web page, continuous integration environment, or server-side environments.

Figure 3.10 shows the failing of a test case that is trying to verify the loading on the JavaScript library. Figure 3.11 shows two passing test cases that verifies the loading of the

necessary JavaScript library and the loading of the store with data in response to the API call. This test gets the *PatientStore* which is asynchronous from the *PatientController* asserts that the store was successfully retrieved waits for the store to complete loading -- see the "waitFor" function. This store auto loads data when its constructed.

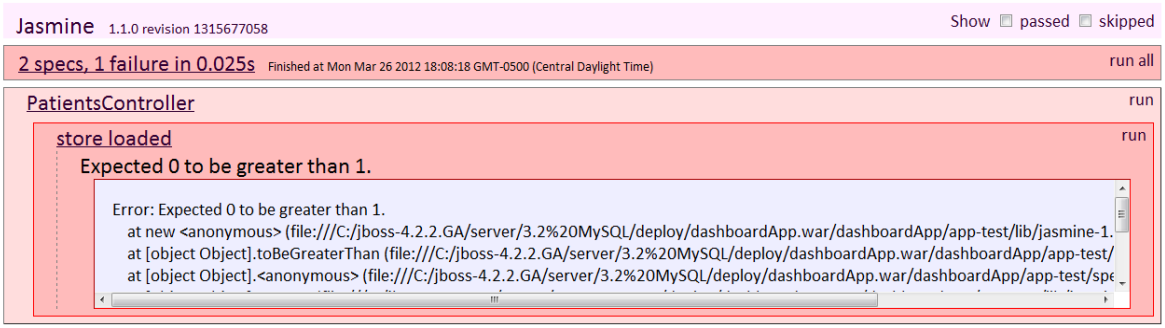

Figure 3.10 Failing Test Cases

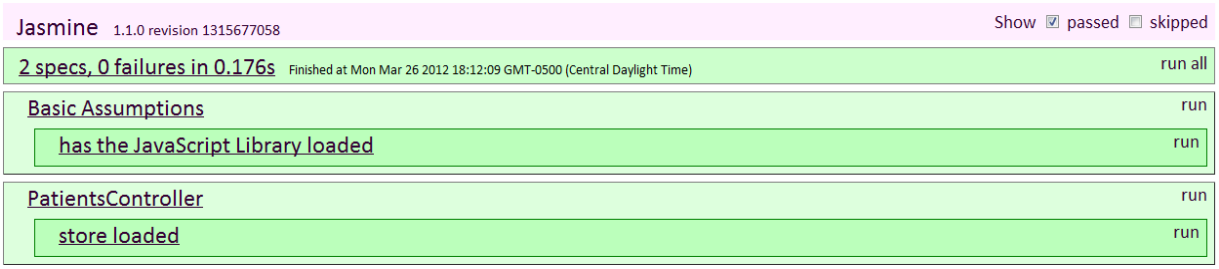

Figure 3.11 Passing Test Cases

3.6.2. Functional Testing

The purpose of the functional testing in the Dashboard Application is to verify that the application behaves correctly from the user perspective and functions according to the use cases of the requirements specified in the application. The function test must determine if each component or business event: performs in accordance to the specifications, responds correctly to all conditions that may be presented by incoming events / data, moves data correctly from one

business event to the next (including data stores), and that business events are initiated in the order required to meet the business objectives of the system.

Table 3.2 and Table 3.3 describe two test cases to validate the business use cases of the Dashboard Application.

Table 3.2 Test Case 1: Logging Into Dashboard Application

Test Case ID	**DashboardApp-TestCase-01**
Test Description	Verify logging into the Application with username & password
Revision History	3/3/2012 Rev.#1600
Function to be Tested	Logging in with proper credentials
Environment	Windows 7 using Firefox version 10.0.2
Test Setup	N/A
Test Execution	1. Type the URL to the Dashboard Application 2. Verify the Login page loads 3. Type the valid username 4. Type the valid password 5. Click the Login Button
Expected Result	Dashboard Application correctly logs you into the system with valid username and password
Actual Result	Pass

Table 3.3 Test Case 2: Patients' Scheduled OR Name and Scheduled Time

Test Case ID	**DashboardApp-TestCase-02**
Test Description	Verify Patients' Scheduled OR name and Scheduled Time is not empty
Revision History	3/3/2012 Rev.#1600
Function to be Tested	Patients' record need to have Scheduled OR location and Scheduled Time.
Environment	Windows 7 using Firefox 10.0.2
Test Setup	N/A
Test Execution	1. Login to the Dashboard Application with valid username and password 2. Check the Scheduled Time and Scheduled OR Name column corresponding to each patient
Expected Result	Dashboard Application displays the Scheduled Time and the Scheduled OR Name for each patient
Actual Result	Pass

3.6.3. Performance Testing

Performance testing is a type of testing intended to determine the responsiveness, reliability, and scalability of a system under a given workload. For the Dashboard Application performance testing involves the following activities:-

- identify the testing environment and tools,
- define the responsiveness criteria or bench mark from the business use case,
- run and monitor the test to validate the use case and
- re-execute the test as needed.

The test tool used is called *dynaTrace AJAX* edition which traces both JavaScript executions as well as all calls to the DOM, perhaps the most performance critical component in the browser. In addition, the performance test tool automatically generates reports to tune browser caching, reduce network roundtrips and optimize server-side request times.

Bench mark: As a Dashboard Application User, I need to be able to see the application loaded on supported browsers (see Appendix A) in under 2 seconds [15]. Supported browsers are Firefox version 10 and above, Internet Explorer version 9 and above, Web Kit browsers such as Chrome 17 and Safari 4 or higher.

Actual Output: As evidenced in Figures 3.9 and 3.10, performance testing using dynaTrace AJAX edition revealed that the Dashboard Application gets fully loaded in 2.77 seconds. The bench mark use case dictates 2 seconds [15].

Next Step: The informative output provides areas in the application that need improvement. Fine tuning of the application such as minifying the JavaScript libraries and refining the filtering in the API call is the necessary step to follow in order to achieve the bench mark. Minifying javascript is the process of removing unnecessary characters such as comments, unneeded white space, newline and tab from code to reduce its size thereby improving response time performance. The load times will be improved because the size of the downloaded file is reduced.

3.7. The Scenarios Simulations

In a typical healthcare facility, surgical cases are usually prioritized and issued one business day in advance of the planned surgeries. Scheduled patients are notified a day early by phone to arrive 1-2 hours ahead of their scheduled start-time. The OR charge nurses use this perceived final version of the schedule to make room assignments and determine staffing needs for the day [5]. On the day of the surgery, this "final" version quickly becomes obsolete as time progresses. The schedule requires several revisions due to cancellations, late arrivals, new additions and changes in procedure. Revisions to this schedule get done at the admission desk but never get communicated to patients or surgeons who have already made their plans on previous day scheduled cases [6]. Expectations were therefore based on the "final" schedule issued the previous day.

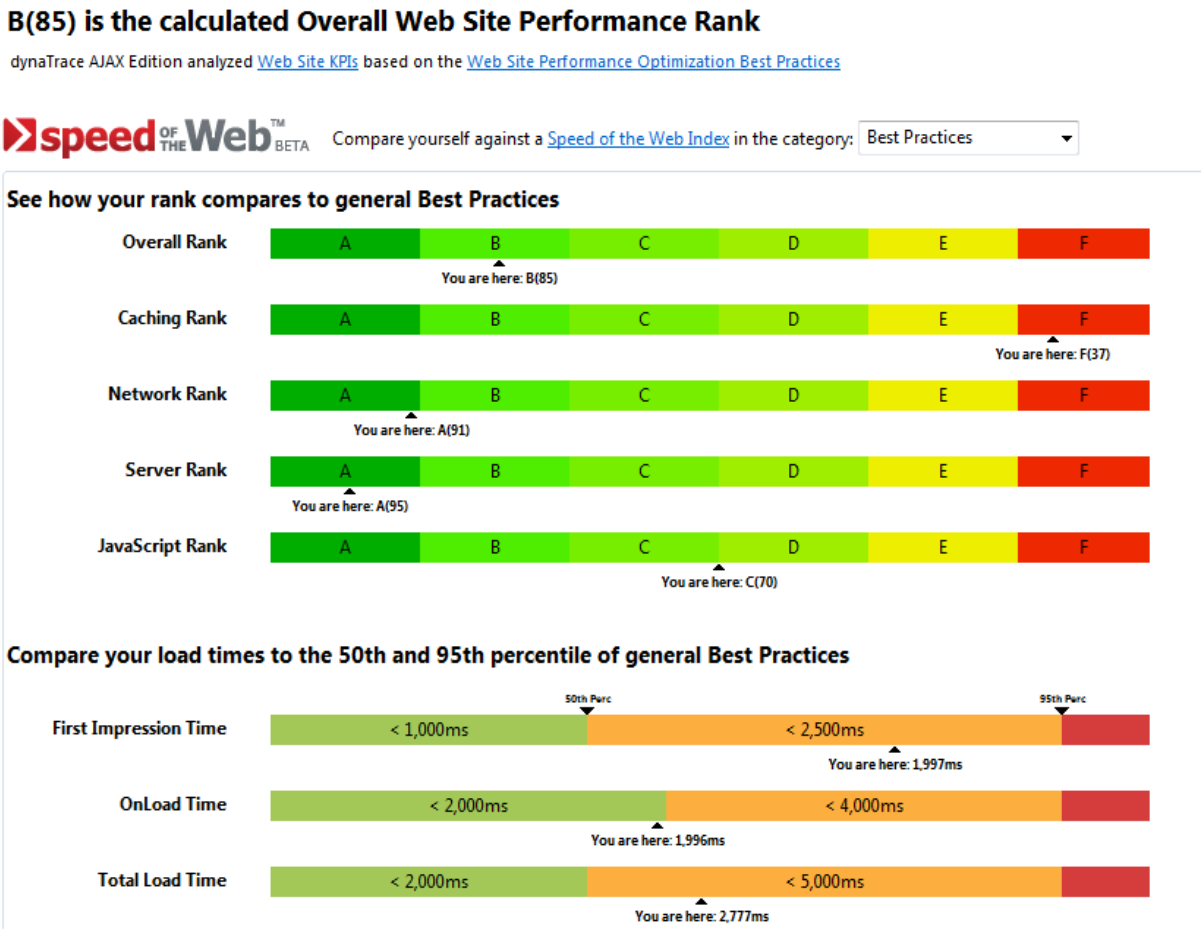

Figure 3.12 Overall Performance Ranking of Dashboard Application

This Dashboard Application will change the above work flow scenario by continuously updating the information about the scheduled and arrived patients' pre-surgical status and presenting it visibly on monitors similar to the airport display screens with current flight status. The application continually polls for updated current tag locations as the movements of tags affixed on patients are sensed at room level. The Dashboard Application also provides a visual way to easily spot the patient progress with regards to tardiness for their surgery.

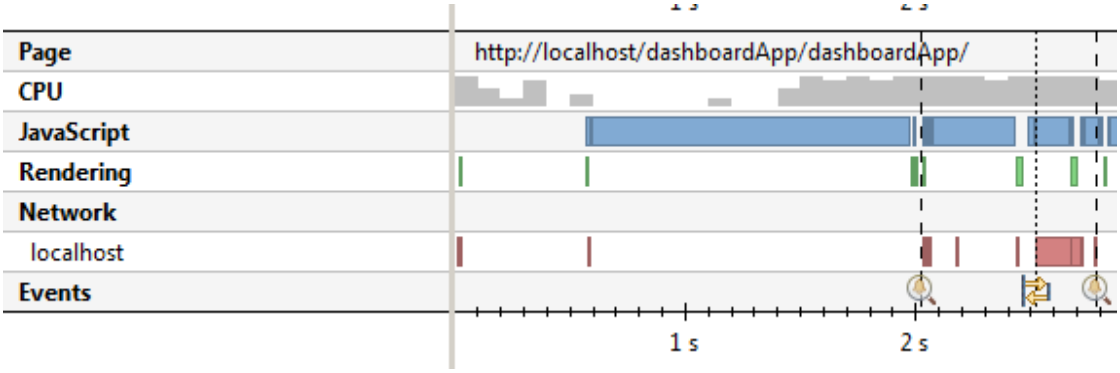

Figure 3.13 Timeline View of the Page Loading Activity

The 3 stage color indicator shows **Green** if the patient's progress is within the time allotted for it; **Yellow** if the patient progress is at the risk of approaching tardiness and **Red** if the patient progress has been delayed beyond the scheduled surgery time and is officially tardy. The acceptable duration of time a patient stays within each milestone is pre-determined by the healthcare facility.

When scheduled patients arrive for surgery and before the start of their operation, the following sequence of events typically occurs to ensure preparedness for surgery:-

1. **Admission** – patient arrives, patient's personal information is verified, schedules are confirmed and tag is affixed.
2. **Waiting Area** – patient waits for initial contact with pre-op nurse
3. **Lab** – where necessary, certain pre-operative tests (like blood tests, EKG, Chest X-ray, etc.) are performed, and patient's physical status is rated.
4. **Pre-operative Holding Area** –
    a. **Paperwork** - If all tests results are satisfactory, the patient signs a consent form and is given a surgical clearance.

b. **Changing** - Patient changes out of his or her street clothes and is asked to confirm the details of his or her surgery.

c. **Vitals** - A set of vital signs is recorded, a peripheral IV line is placed, and pre-operative medications (antibiotics, sedatives, etc.) are given.

d. **Anesthesiologist** performs a brief physical examination; takes a patient history; and obtains information regarding medication used on a regular basis, drug allergies, and prior adverse reactions to anesthesia. This information helps the anesthesiologist select the most suitable anesthetic agents and dosages to avoid complications.

e. Surgeon visit – A surgeon visit in preoperative areas is becoming more and more obligatory whether to answer any last minute questions from patients or their family members or to provide assurance to the patient.

f. **Optional visit by the Lab technicians** if test results appear abnormal, resulting in time stolen from scheduled OR start-time.

g. **Optional visit by a Radiologist** when necessary, resulting in time stolen from scheduled OR start-time.

5. **Transportation –** patient ready to be wheeled into the assigned operating room.

6. **Arrival in OR suites**

There are several scenarios where OR staff need to be ready for a change of schedules in start-times. Patient cancellation is one such case. Cancellation can occur before the patient even arrives at the facility or at any point in time during the pre-operative procedures. Another scenario is case delay that can occur when a patient arrives late for their scheduled time or

anomalies in the pre-operative stages occur that run past the allotted time and hence, causing tardiness.

Table 3.4 describes various potential scenarios that a patient could undergo and how the Dashboard Application could highlight the issues associated with each scenario.

Table 3.4 Simulation of Six Scenarios using Dashboard Application

Simulation	Description
Simulation 1	**Jane Doe arrives on time and gets wheeled to her scheduled OR on her scheduled time.** • Jane Doe arrives at Admissions on the recommended prep time • She gets registered and receives her tag • Her movement from one location to the next gets sensed and tracked as she moves through the waiting area, pre-operative holding room and transportation center. The Dashboard Application displays the current location and; time entered at that location along with the calculated time remaining before her scheduled surgery start-time. Since Jane's procedure is progressing as planned, the color indicator remains green throughout the pre-operative stages. • OR staff monitors the Dashboard Application mounted on the wall and gets ready for her because they are aware of her every step.
Simulation 2	**Becky Wheeler arrives late but makes up the time in pre-operative area because of the Dashboard Application.**
	• Becky Wheeler arrives later than the recommended prep time for her

Table 3.4 (continued)

	scheduled surgery. • She gets registered and receives her tag at Admissions. • Pre-operative holding area staff notices the Yellow color from the Dashboard Application's dynamic color indicator that Becky is in danger of being tardy for her scheduled surgery start-time, so the Pre-OP staff will try to facilitate her procedure. • OR staff also notices the late arrival and closely monitors Becky's progress as she advances in her pre-operative stages. Her location gets tracked as she moves through the waiting area, pre-operative holding room and transportation center with the time stamp at each location. • Each of her location changes get broadcasted, along with the time spent at that location and the time remaining before her scheduled OR surgery start-time. • Becky's color turns green as she progress to the transportation center to be wheeled to her scheduled OR suite.
Simulation 3	**Mary Pop cancels her schedule surgery for the day. Dashboard Application reflects that change so OR staff can determine the next step.**
	• Mary Pop was scheduled for surgery today in OR at 09:00 AM. • OR charge nurse observes that Mary Pop doesn't show in the

Table 3.4 (continued)

	Dashboard Application when she was supposed to. OR charge nurse communicates with admission and releases Mary Pop's OR for new patients or early arrivals.
Simulation 4	**Bob Smith starts on time but his surgery is cancelled in pre-operation. Dashboard Application communicates this information so the OR suite can be re-allocated to another patient.** • Bob Smith arrives ahead of the recommended prep time for his scheduled surgery. • He gets registered at the admission desk and receives his tag to track his movement. • He moves from the Waiting Area to the Lab Area because of his pre-existing condition. • Unfortunately, he receives abnormal lab test results that disqualify him from being a candidate for surgery on that day. • The time lapse at the lab indicates that Bob is tardy for his scheduled start-time. The Dashboard Application monitor color indicator turns red to alert the OR staff of the situation. • OR staff notices the red color indicator showing the time and the place Bob is delayed. . • OR charge nurse can release the OR room designated to Bob for an early arrival patient and reschedule Bob to a different OR at a later time or negotiate with admissions for a new and unscheduled patient.

Table 3.4 (continued)

Simulation 5	**Aimie Yung arrives late and as a result she is in danger of being tardy for her scheduled surgery. Dashboard Application tracks her every step in the process, allowing the OR staff to decide the next step.** • Aimie Yung arrives at the admission desk with only a few minutes prior to surgery, leaving no room for pre-operative procedures. • Dashboard Application calculates her time of arrival against the scheduled time and puts Aimie Yung in the yellow zone starting at the admission area. • Dashboard Application keeps track of Aimie Yung's location and her current time in the pre-operative area while changing the color indicator to red as she surpasses her scheduled surgery start-time. • OR staff notices the red color indictor showing the time and place Aimie Yung has been delayed. • Glancing at the Dashboard Application reveals where an OR room may be available • OR charge nurse can release the OR room designated to Aimie Yung for an early arrival patient and reschedule her to a different OR at a later time or negotiate with admissions for a new and unscheduled patient.
Simulation 6	**John Doe, unscheduled and new surgery case (perhaps on waiting list or clerical entry error) receives service because of Simulation #3**

Table 3.4 (continued)	
	or Simulation #5 cases.
	• John Doe arrives at the admission desk of the healthcare facility. • John was not listed in the schedule that was issued a day early because of clerical entry error. • He is marked as a new patient. From the Dashboard Application, OR staff and the registration staff can see that there are openings for surgery in place of Simulation #3: Mary Pop or Simulation #5: Aimie Yung. • OR charge nurse can negotiate to provide service for John Doe. • If so, he gets a tag and the process of tracking his progress and preparedness begins.

3.8. Calculating Patient Tardiness and Simulation Logic

Patients are required to undergo the following pre-operative stages for the purpose of this simulation: Admission, Waiting, Lab/Exam, Pre-operation and Transport. Figure 3.11 illustrates the series of decision making processes involved in calculating the patient's status with regards to their scheduled time.

The maximum allowed time for each stage is pre-defined for this specific simulation (5 minutes for Admission, 15 minutes for Waiting Area, 30 minutes for lab, 50 minutes for pre-operative procedures, 10 minutes for transport). As a patient arrives at the location, the time of arrival at the particular location is checked against its progress start time which is calculated as

the schedule time minus the sum of the maximum allowed time for each location. Patient is tardy if time entered at location is greater than progress start time and a tardiness icon indicator along with location of tardiness will be displayed at the Dashboard Application. If time entered at location is equal to progress start time, then a warning icon indicator and message will be displayed on the Dashboard Application.

Warning time indicator for each pre-operative stage is calculated as 1/3 of the maximum allowed time at that location. The progress end time at each location is calculated as the difference between scheduled time minus the sum of the maximum allowed time each location minus duration at that location. The application continually calculates the patient's time markings every sixty seconds. If the current time is equal to or exceeds the progress end time, then the patient is tardy. If the current time is less than the progress end time and the warning time, then the patient is progressing according to the schedule. But if the current time is after the warning time but before the progress end time, the patient is approaching tardiness so the warning indicator will be displayed on the Dashboard Application.

The software could archive the real-time transition times between locations for each patient. This information could then be analyzed to create better metrics for the time allocations for the phases. This aggregate information could also be used for other analytic needs, such as identifying problematic processes.

3.9. Simulation Clock Management

The purpose of the simulation is the modeling of the Dashboard Application system which is dynamic and event driven. The application's behavior is expressed as a function of time with

operations that involves sequence of activities which changes the state of the variables (i.e. patient's status).

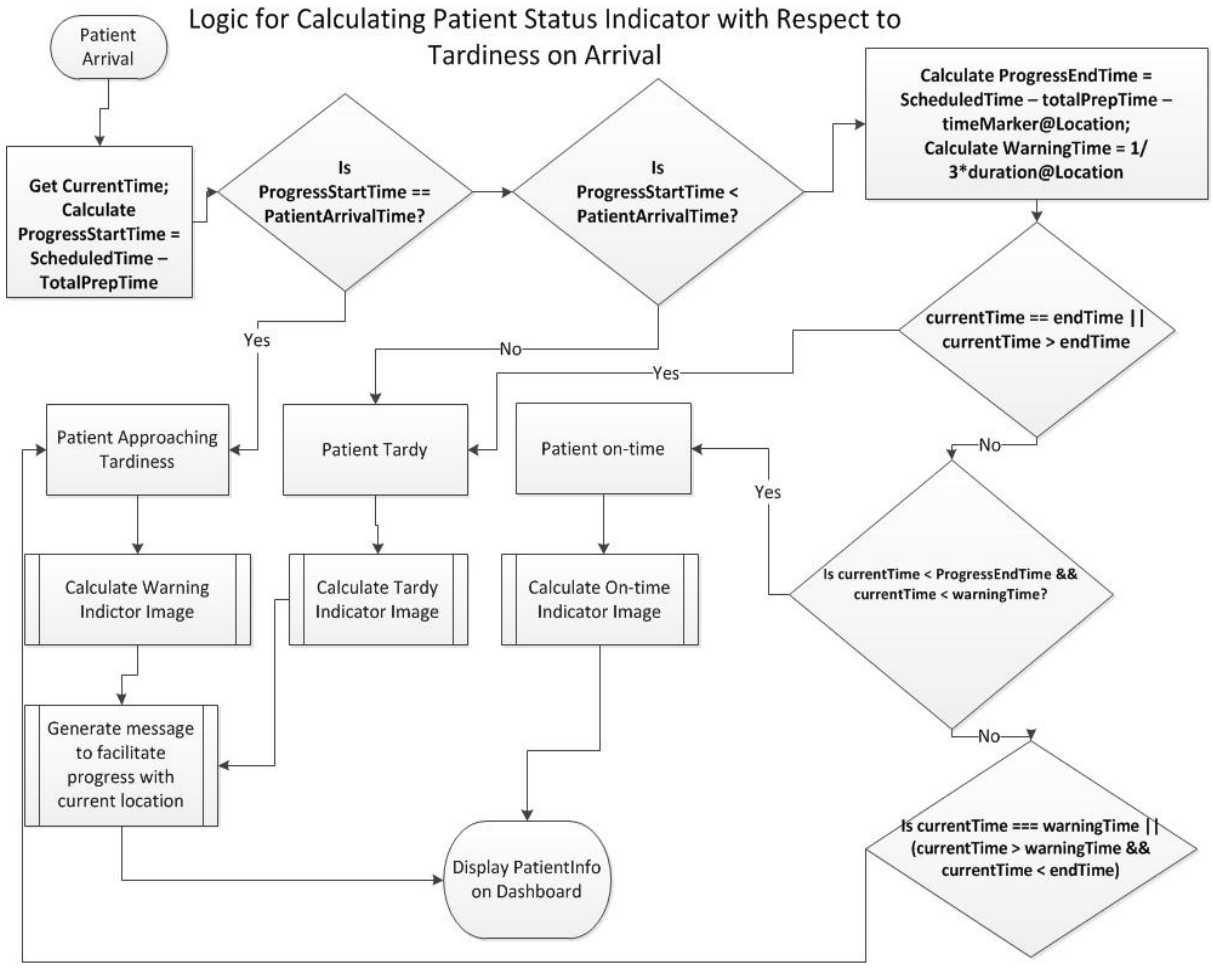

Figure 3.14 Logic for Calculating Patient Status at Each Pre-operative Stage

The Dashboard Application is a discrete-event model; when an arrival event occurs, it causes a change in the state variable that represents the status of patient, length of stay at location, time remaining before surgery and the number of patients displayed in the application. This patient status state variable and any other only change its value when an event occurs, i.e., at discrete instants.

From the operational point of view, discrete-event simulation consists of a sequence of events, ei, occurring at event times, ti. The Patient Controller class is the simulation executive that ensures listening to location movement updates occur in sequence of 60 seconds interval, thereby, incrementing the simulation clock. The application implements both fixed time increment (every 60 seconds) and variable time increment (as patient moved event occurs) simultaneously to change the time of the simulation clock. These two approaches to handling time, periodic scan and event scan, are simple approaches to simulating parallelism. For the fixed time increment, an API call is instantiated every 60 seconds to determine any change occurred during that interval. If any change occurred, it is simulated and the execution of the method renderCalculatedImage is invoked to determine the patient status and display appropriate

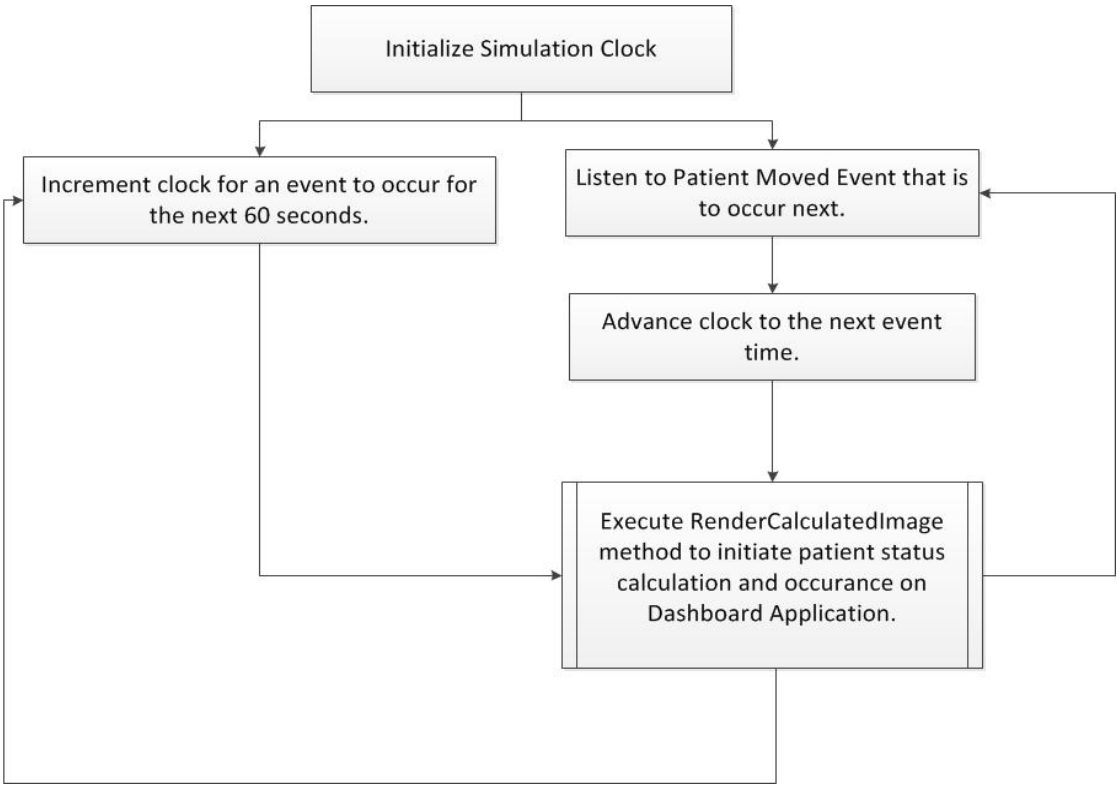

Figure 3.15 Flow Chart for Simulation Clock Management

response. The simulation clock is then advanced to another time unit, and the cycle is repeated. The Patient Controller class shown in Figure 3.16 is also subscribed to listen to a patient moved event from the API. When a patient is advanced to the next stage in the pre-operative stage, the simulation clock is incremented, the simulation method to calculate the patient status at that location is implemented and the whole cycle is repeated.

The Dashboard Application provides the necessary visibility for capturing potential tardiness, by displaying color coded indicator for impending transitions. Using the color coded icons at each stage of the pre-operative process; staff can easily identify which patients need to be given priority. Patient with red are behind schedule and patient with yellow are soon to be behind.

```
Ext.define('dashboardApp.controller.PatientsController', {
        extend: 'Ext.app.Controller',
        requires: ['dashboardApp.events.EventManager'],
        refs: [
            { ref: 'patientList', selector: '#dashboard-patient-list' }
        ],
        views: ['patient.List'],
        stores: ['PatientsStore'],
        init: function() {
        var me = this;

         var failureFunction = function(response) {
        Ext.Ajax.request({
            url: '/api/2.0/rest/events/queue.json',
            method: 'POST',
            params: { 'long-polling-seconds': '15', filter:"event-type eq 'moved'", "word-separator": "camel" },

            success: function(response) {
                 try {
                        var returnObject = Ext.decode(response.responseText);
                    } catch(e) {

                        failureFunction(response);
                        return;
                    }
                    var uri = returnObject['event-queue'] != undefined ? returnObject['event-queue'].uri : '';
                    var readFunction = function() {

                    readFunction();
```

Figure 3.16 Code Snippet for Patient Controller Class

4. RESULTS AND CONCLUSION

4.1. The Results

The Dashboard Application was implemented with six sample scenarios described in Table 3.4 with varying scheduled surgical times to account for a potential step in tardiness. Tardiness was measured as a lapse of time incurred from a scheduled surgical start-time. If a case was scheduled to enter its designated OR at 10:00 AM but enters at 10:40 AM, then that case is tardy by 40 minutes. Five required pre-operative stages (Admission, Waiting, Lab/Exam, Pre-operation and Transport) were pre-defined for the purpose of this simulation. The corresponding maximum allowed duration of stay for each pre-operative stages were also pre-determined. As the patient arrives at admission desk and receives a tag, time remaining is calculated from scheduled time minus the total of the maximum duration. Patient was on time if the time entered at that location was before scheduled the time remaining, tardy if the time entered is after the time remaining and approaching tardiness. Tardiness was measured when the patient arrives later than the designated time or the patient stays at one of the pre-operative stages past the allocated duration of time for that location. Patients automatically disappear from the Dashboard Application when they are transported to the entrance of the OR Suites. Only OR patients scheduled for the day were filtered to appear on the Dashboard.

This Dashboard Application continuously updates the information about the arrived and patients' pre-surgical status and presented it visibly on monitors similar to the airport display screens with current flight status. The application also continually polled for updated current tag locations as the movements of tags affixed on patients are sensed at room level.

The Dashboard Application provided a visual way to easily capture the patient progress with regards to tardiness for their surgery. The 3 stage color indicator showed **Green** if the patient's progress is within the time allotted for it; **Yellow** when the patient progress was at the risk of approaching tardiness and **Red** if the patient progress had been delayed beyond the scheduled surgery time and was tardy.

The Dashboard Application:

- Provided visibility to the communication aspects of patient preparedness for surgery so that hospital staff don't have to use traditional modes of communication (face-to-face and telephone).
- Provided real-time and accurate information to assist the charge nurse to make proactive real-time decisions.
- Helped capture potential areas for tardiness.

Without the use of automated Dashboard Application, the percentage of traditional modes of communication (face-to-face and telephone/intercom) as identified by Moss et al. was significantly high. With the automated implementation of the Dashboard Application, the real-time nature of the communication and the availability of the application on mounted devices for all OR-related hospital departments would reduce the high need for the face-to-face communication. The telephone/intercom communications can be reserved negotiating time for a new case (patient), negotiating a change in case time due to late arrival, cancel a case, assigning a case to a particular room or assigning a particular staff to a case.

4.2. The Limitations

Success of the Dashboard Application depends deeply on the people using it. Although not a major constraint, the use and interpretation of the Dashboard Application requires a learning curve for the staff. A compulsory one hour training session on the use and interpretation of the Dashboard Application needs to be offered to the staff. The adoption of this application and new way of communicating will be most cost-effective when combined with Lean and Six Sigma processes to improve organizational culture and prompt decision making [9].

The Dashboard Application uses room-level accuracy to pin point the location of patients with respect to their progress while undergoing pre-operative procedures. As a result, the Dashboard Application assumes patients to arrive at the designated pre-operative rooms.

Besides the fact that this paper is based on simulations and not real live scenarios, the Dashboard Application inherits the limitations of the RTLS technology as described in Table 4.2.

Table 4.1 Summary of RTLS Limitations

Location Accuracy	The zone level and room level accuracy of location sensors is dependent on: • The RTLS technology deployed in the healthcare facility (Wi-Fi, Infrared, hybrid system, etc.) • Robustness of the location engine algorithms to determine the precise location of the tag being tracked

Table 4.1 (continued)

	• The level of integration between the RTLS and the other APIs • The way tags that are not reporting and not locatable are handled. Some networks may already have capacity issues, so adding traffic to the network with a tracking system may affect the responsiveness of the tracking system.
Battery Life	The frequency of tag location updates may have implications for the number of tags that can be deployed and the battery life of the tag. In typical applications, systems can track thousands of tags simultaneously. Depending on the hardware vendor, the average battery life of tags is quite variable. The RTLS batteries are replaceable and they chirp the signals of low battery indicator. In cases where the low battery status of a tag is not identified on time, the patient carrying the tag will not be tracked.
Maintenance Efforts	The main factor for ongoing success of RTLS deployment involves maintenance considerations. These includes: replacing batteries in the tags, replacing tags or location sensors that aren't

Table 4.1 (continued)

	functioning properly, adding new tags and location sensors, understanding the needs to tweak location sensors to achieve acceptable results, and monitoring whether all sensors are operational. A service model that is monitored 24/7 that provides ongoing system maintenance and data back-up must be in place.
Security & Privacy Issues	Due to the silent and invisible character of the technology that enables real-time locating of people over the air, an RTLS introduces unique challenges to security and monitoring performance [16]. Monitoring and measuring the performance of an RTLS can be challenging, especially with all the tags, location sensors, technology intricacies, and interdependencies within an organization's infrastructure. Confidentiality is, in general, a fundamental security requirement for most organizations. However, due to the shared nature of media used for an RTLS, confidentiality is a more difficult security requirement to meet. Malicious entities may gain unauthorized access to the network through a wireless network made available for the RTLS.

4.3. The Conclusion

This paper has highlighted the valid need for timely and accurate communications between OR staff and other departments in the healthcare facility with regards to the preparedness of patients for surgery. To better understand the reliable information needs in any given healthcare facility OR suites, this paper built up on previous studies that revealed the lack of visibility to the timely and accurate information for OR staff. Real-time and accurate information is necessary for OR charge nurses to make informed decisions when negotiating time for a new case (patient), negotiate a change in case time due to late arrival, cancel a case, assign a case to a particular room, assigning particular staff to a case.

This paper demonstrates that automating aspects of the information delivery mechanism regarding patient preparedness provide timely updates and decreased interruptions. Delivering visibility aims at equipping healthcare workers with real-time knowledge to change the outcome of their environment and effectively communicate the potential for start-time tardiness.

The Dashboard Application was used as a tool that exhibited the benefits of automating the process of tracking patients' locations to improve communication and workflow for OR staff. By displaying the real-time location of a patient and the time at their current location along with the remaining time left for scheduled surgery, the Dashboard Application outlined the different scenarios where the need to access real-time information was critical to facilitate the communication between departments and devise strategies for pointing out and ultimately minimizing the most common time stealers and delays.

The real-time simulation of the previously described six scenarios using the Dashboard Application revealed that RTLS based automating of patient flow, made visible the locations

where delays occur during the patient stages of preparedness and opened the door for multiple possible solutions to fix the problem for the continuation of workflow and to optimize efficiency.

The Dashboard Application provides the necessary visibility for capturing potential tardiness, by displaying color coded indicator for impending transitions. Using the color coded icons at each stage of the pre-operative process; staff can easily identify which patients need to be given priority. Patient with red are behind schedule and patient with yellow are soon to be behind. A future implementation of an automated paging system integrated with the Dashboard Application to alert pre-operative staff would be beneficial to assist expediting the patient movement when they are approaching tardiness. The operating room staff also needs a good training to use this real-time information to effectively communicate with the pre-operative departments to expedite patient progress and mitigate tardiness.

Additional important contributors to improved efficiency are more flexible OR staffing, improved personal accountability, streamlining of procedures, interdisciplinary team work, and accurate data collection [8]. The pursuit of improved efficiency requires a seamless blend of trained people, processes, and technology to deliver the cost, quality, and capacity improvements required for the future.

5. FUTURE IMPROVEMENTS

A number of enhancements could be applied to this Dashboard Application that will improve the automating process of communication between the OR charge nurse and other hospital units, in an effort to reduce tardiness in patient flow and to improve patient satisfaction. The Dashboard Application could be extended to include a mechanism to send alerts and notifications to the OR charge nurse's pager when a patient is at the risk of approaching tardiness (when the color indicator changes to yellow and ultimately to red). This feature would be helpful as an additional way to urge the OR charge nurse to follow up with the pre-operative staff to accelerate the preparedness process with the knowledge that the scheduled time is approaching. The same alert and notification could be sent to the pre-operative unit staff to remind them to facilitate the process or the PACU staff to adjust their schedule accordingly. This feature enhancement will require the admission desk to include with each patient the pager number of the charge nurse or other department staff that will receive the alerts and notifications.

Alternatively, a slightly modified version of the Dashboard Application could be used to update families of patients in the waiting area regarding the progress of their loved ones as they get ready for surgery, when and where they start surgery, at what time they enter recovery at PACU and when they are expected for discharge. This version of the Dashboard Application would take into account patient confidentiality and could display a randomly generated ID in place of the name and photo of the patient.

Another visual pleasing enhancement of the Dashboard Application could be the use of the healthcare facility's floor plan as a map to indicate in real-time the movement of patients

from one room, floor or campus to another. This would further provide the at-a-glance view of the patient flow to OR staff and other interested parties.

Additionally, the daily issued schedule of surgery start-times could be dynamically updated throughout the day to reflect the actual patient flow. Consequently, these revised start-times could be used to aid in predictive modeling analysis and for better patient arrival times.

6. REFERENCES

[1] OECD, "Health at a Glance 2011: OECD Indicators," 2011.

[2] Committee on Quality of Health Care in America, Institute of Medicine, *Crossing the Quality Chasm: A New Health System for the 21st Century*, 1st ed. Washington, D.C., United States of America: National Academies Press, 2001.

[3] E. Litvak, B. Prenney, K. K. Fuda, M. C. Long, and P. McGlinchey, "Improving Patient Flow and Throughput in California," Boston University Health Policy Institute, Boston, Research Study 2006.

[4] A. Macario. (2010, April) Medscape Anesthesiology. [Online]. http://www.medscape.com/viewarticle/719542

[5] F. Dexter and R. E. Wachtel, "Influence of the Operating Room Schedule on Tardiness from Scheduled Start Times," *Anesthesia & Analgesia*, pp. Vol. 108, No. 6, pp.1889-1901, 2009.

[6] R. E. Wachtel and F. Dexter, "Reducing tardiness from scheduled start times by making adjustments to the operating room schedule," *PubMed*, pp. 1902-9, 2009.

[7] J. Moss and Y. Xiao, "Improving operating room coordination: communication pattern assessment," *The Journal of Nursing Administration*, vol. 34, no. 2, pp. 93-100, February 2004.

[8] C. B. Fairbanks, "Using Six Sigma and Lean methodologies to improve OR throughput.,"

AORN J, pp. 86:73-82, 2007.

[9] L. Lingard, S. Garwood, and D. Poenaru, "Tensions influencing operating room team function: does institutional context make a difference?," *Med Educ*, vol. 37, pp. 691-9, 2004.

[10] S. D. Lapierre, C. Batson, and S. McCaskey, "Improving on-time performance in healthcare organizations: a case study," *Health Care Management Science*, vol. 2, pp. 27-34, January 1999.

[11] S. V. Wagenen, "Early RTLS adopters report success with initiatives," *Healthcare IT News*, pp. 2-8, September 2009.

[12] A. Malik, *RTLS for Dummies*, 1st ed. Hoboken, USA: Wiley Publishing, Inc., 2009.

[13] iINTERFACEWARE. (2012, January) HL7 Overview. [Online]. http://www.interfaceware.com/hl7.html

[14] S. M. Lauriat, *Advanced Ajax: Architecture and Best Practices*. Boston: Prentice Hall; 1 ed., 2007.

[15] F. Nah, "A study on tolerable waiting time: how long are Web users willing to wait?," 2004.

[16] L. S. Strickland and L. E. Hunt, "Technology, security, and individual privacy: New tools, new threats, and new public perceptions," *Journal of the American Society for Information Science and Technology*, vol. 56, no. 3, pp. 221–234, 2005.

[17] F. Dexter, A. Willemsen-Dunlap, and J. D. Lee, "Operating room managerial decision-making on the day of surgery with and without computer recommendations and status displays.," *Anesthesia and Analgesia*, pp. 105(2):419-29., August 2007.

APPENDIX A. SOFTWARE TOOLS & TECHNOLOGIES

Table A1 lists the software tools and technologies used in order to accomplish the tasks in this paper and run the Dashboard Application. Programming language selection is based on acquired skills and robustness of the framework for the application.

Table A1 List of Software Tools & Technologies Used

Web Server & Web Service	JBoss Application Server & RESTful
Web Browser	Fire Fox 9, Internet Explorer 9, Safari and Chrome and Opera
Graphic User Interface	Microsoft Visio, Microsoft Power Point, Photoshop CS 4
Development Environment	IntelliJ 11
Web Rendering Tool	HTML5, CSS3, SASS, Web Kit
Testing	Jasmine unit testing framework
Operating System and Database	The system was running on windows 7, with MYSQL server running
Web Technologies & Languages	HTML 5, CSS 3, SASS, JavaScript (ExtJS framework), JSON, Java
Mobile Access	Safari, Chrome Web Kit
Hardware Requirements	Any configuration of hardware that can support Windows, HTTP protocol and internet connection, and RTLS.
RTLS Integration Technologies	Wi-Fi, ZigBee

APPENDIX B. SOURCE CODE – PATIENT CONTROLLER

```
Ext.define('dashboardApp.controller.PatientsController', {
        extend: 'Ext.app.Controller',
        requires: ['dashboardApp.events.EventManager'],
        refs: [
                { ref: 'patientList', selector: '#dashboard-patient-list' }
        ],
        views: ['patient.List'],
        stores: ['PatientsStore'],
        init: function() {
    var me = this;
    var failureFunction = function(response) {
    this.consecutiveAjaxErrors += 1;
    if (this.consecutiveAjaxErrors < 5) {
       new Ext.util.DelayedTask(function() {
       }).delay(5000);
    } else {
       if (this.consecutiveAjaxErrors) {
          //console.log("Communication Error");
          Ext.Msg.show({
             title:"Communication Error",
             msg:"Unable to communicate with server, please refresh this screen (F5) ",
             icon: Ext.Msg.ERROR,
             buttons:Ext.Msg.OK});
       }}
```

```
};
Ext.Ajax.request({
    url: '/api/2.0/rest/events/queue.json',
    method: 'POST',
    params: { 'long-polling-seconds': '15', filter:"event-type eq 'moved'", "word-separator": "camel" },
    success: function(response) {
     try {
        var returnObject = Ext.decode(response.responseText);
    } catch(e) {
    failureFunction(response);
    return;
  }
  var uri = returnObject['event-queue'] != undefined ? returnObject['event-queue'].uri:";
var readFunction = function() {
    Ext.Ajax.request({
      url: uri + '.json',
      method: 'GET',
      headers: {
        'Content-type': 'application/x-www-form-urlencoded'
      },
      success: function(response) {
        if (!(response && response.responseText)) {
          failureFunction(response);
          return;
        }
        try {
```

```
                    var eventList = Ext.decode(response.responseText);
                } catch(e) {
                    failureFunction(response);
                    return;
                }
                this.consecutiveAjaxErrors = 0;
                Ext.each(eventList['event-queue'].events.value, function(newEvent) {
            //Get the event from event Queue and send it to the patient moved event function
            //for processing
                    me.patientMovedEvent(newEvent);
                });
                readFunction();
            },
            failure: failureFunction
        });
    };
    readFunction();
},
failure:  function(response) {
    me.consecutiveFailures = me.consecutiveFailures+1;
    if (me.consecutiveFailures > 5) {
        Ext.Msg.alert('Communication Error', 'Unable to communicate with server, please attempt to refresh this screen (F5)');
        //console.log("dashboardApp.events", response.statusText + " error creating event queue.  Giving up.  Response: " + response.responseText);
    } else {
        new Ext.util.DelayedTask(function() {
        }).delay(5000);
```

```
            console.log("dashboardApp.events", response.statusText + " error creating event queue. Automatically restarting.  Response: " + response.responseText);
                                        }
                                    }
                                });
    },
        initListeners :function() {
                this.serverListen("core.moved", this.doSomething,this);
    },
    serverListen: function(eventCode, handler, scope) {
        if (!this.subscriptions) {
            this.subscriptions = [];
        }
        this.subscriptions.push(dashboardApp.events.EventManager.subscribe(eventCode, handler, scope));
    },
    closeListeners: function() {
        Ext.each(this.subscriptions, function(subscription) {
            dashboardApp.events.EventManager.unsubscribe(subscription);
        });
        this.subscriptions = undefined;
    },
    patientMovedEvent: function(event) {
                this.getPatientList().updateRecord(event);
}
});
```